
INFORME DE PRÁCTICAS

Repositorio de proxecto: <https://github.com/JuniorParadelo/VVS>

Participantes no proxecto:

Brais Álvarez Sanchez

Junior Paradelo Álvarez

Miguel Sande García

Validación e Verificación de Software

Índice

1	Descripción do proxecto	3
2	Estado actual	3
2.1	Compoñentes avaliados	3
3	Especificación de probas	4
3.1	JUnit	4
3.2	Pitest	4
3.3	JMeter	4
3.4	JUnit-QuickCheck	4
4	Rexisto de probas	4
4.1	JUnit	4
4.2	FindBugs	4
4.3	PITest	5
4.4	JMeter	5
4.5	JUnit-QuickCheck	5
4.6	GraphWalker	5
4.7	Cobertura	6
5	Rexistro de erros	6
5.1	FindBugs	6
6	Estatísticas	7

1. Descrición do proxecto

Sinxela aplicación web dun portal de apostas online. Onde se pode apostar a distintos eventos deportivos.

2. Estado actual

Persoas responsables do desenvolvemento: Brais Álvarez Sanchez, Alejandro Sanchez Sanchez, Miguel Sande García.

Persoas responsables do proceso de probas: Brais Álvarez Sanchez, Junior Paradelo Álvarez, Miguel Sande García.

Lista de funcionalidades:

- Realizar aposta: permite a un usuario realizar unha aposta sobre un evento.
- Crear evento: permite a un administrador crear un evento deportivo con todos os seus detalles.
- Buscar evento: un usuario pode buscar distintos eventos mediante nome, categoría ou ambas las dos.
- Crear aposta: un administrados pode xerar unha aposta sobre un evento que aínda non empezara, pode ser única ou con varias gañadoras.
- Crear usuario: permite o rexistro de un novo usuario.

2.1. Compoñentes avaliados

1. UserService:

- funcionalidades: rexistro de novos usuarios e login na aplicación.
- nº de probas: 13
- porcentaxe superada: 100 %
- *mais info no directorio /doc*

2. EventService:

- funcionalidades: permite a creación de eventos e categorías, ademais da sua búsqueda.
- nº de probas: 8
- porcentaxe superada: 100 %
- *mais info no directorio /doc*

3. BetService:

- funcionalidades: permite a creación de apostas por parte de un usuario e a sua búsqueda.
- nº de probas: 4
- porcentaxe superada: 100 %
- *mais info no directorio /doc*

4. BetTypeService:

- funcionalidades: permite a creación de opcións sobre as que os usuarios poden apostar e actualizar o seu estado.
- nº de probas: 7
- porcentaxe superada: 100 %
- *mais info no directorio /doc*

3. Especificación de probas

Ferramentas utilizadas:

3.1. JUnit

Empregamos JUnit para facer probas de unidade da parte modelo da práctica. Tiñamos unha base xa feita e fumos mediante a aplicación de valores frontera, facendo os test necesarios que non tiñamos ben pensados.

3.2. Pitest

Con PITest facemos probas de mutación de código. Con elas conseguimos que algunhas expresións do código sexan mutadas por outras expresións e ver como afectarían eses cambios nela.

3.3. JMeter

JMeter empregámolo para os test de estrés. Grazas a esta ferramenta podemos someter a diferente carga a aplicación e ver como se comporta. Podemos ver nun gráfico como avanza o rendemento dela e poder ver os posibles fallos para correxilos.

3.4. JUnit-QuickCheck

Con JUnit-Quickchek podemos darlle valores aleatorios a diferentes parámetros dos métodos da aplicación e ver si o resultado diso está controlado ou si provocaría erros graves.

4. Rexisto de probas

4.1. JUnit

Para facer probas en JUnit, empregamos as anotacións útiles para os diferentes casos de probas (@Test, @After, @Before...) así como indicando en cada test que poida devolver unha excepción. Comprobamos con assertTrue() si os resultados devoltos eran os esperados, dando así a proba como finalizada.

<https://github.com/JuniorParadelo/VVS/blob/master/doc/TESTUNIDAD.pdf>

4.2. FindBugs

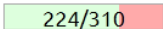
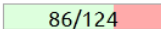
Instalando o plugin de FindBugs dende o propio eclipse, podemos facer mediante o dobre click na raíz do proxecto, pódese seleccionar a opción FindBugs » FindBugs e o plugin checkea tódalas clases que haxa no proxecto atopando os bugs do mesmo. Atopamos 1 bug, con moita gravidade e solventámolo coa propia axuda que xeraba a ferramenta.

4.3. PITest

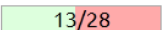
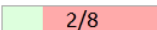
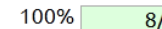
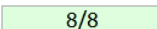
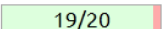
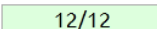
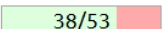
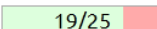
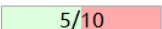
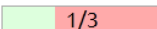
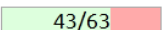
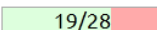
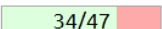
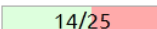
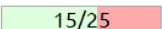
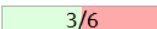
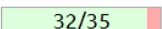
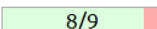
Metendo a dependencia no pom.xml raíz, podemos utilizar a mutación de código automático das expresión que queiramos. Se non lle especificamos ningunha, PITest fará as probas de mutación por defecto. Tivemos algún que outro problema xa que non dabamos excluído algúns test da capa web que non queríamos probar. Para execución dos test utilizamos *mvn org.pitest:pitest-maven:mutationCoverage*.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage	Mutation Coverage
20	72% 	69% 

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage
es.udc.pojoapp.model.bet	2	46% 	25% 
es.udc.pojoapp.model.betType	2	86% 	100% 
es.udc.pojoapp.model.betservice	2	95% 	100% 
es.udc.pojoapp.model.bettyeservice	3	72% 	76% 
es.udc.pojoapp.model.category	2	50% 	33% 
es.udc.pojoapp.model.event	2	68% 	68% 
es.udc.pojoapp.model.eventservice	4	72% 	56% 
es.udc.pojoapp.model.option	1	60% 	50% 
es.udc.pojoapp.model.userprofile	2	91% 	89% 

Report generated by [PIT](#) 1.0.0

4.4. JMeter

Empregramos a aplicación de JMeter, creando tests indicando o número de peticións simuladas, o tempo entre as peticións, etc... Podemos poñer diferentes grafos cos que ver en detalle diferentes parámetros como o rendemento a medida que as petición son incrementadas, os tempos de resposta, etc..

4.5. JUnit-QuickCheck

Introducimos as dependencias no pom.xml raíz e fixemos novos test utilizando os métodos de valores aleatorios proporcionados pola ferramenta. Con eses test, podemos ver qué sucede cando os valores introducidos nos métodos son aleatorios ou non válidos.

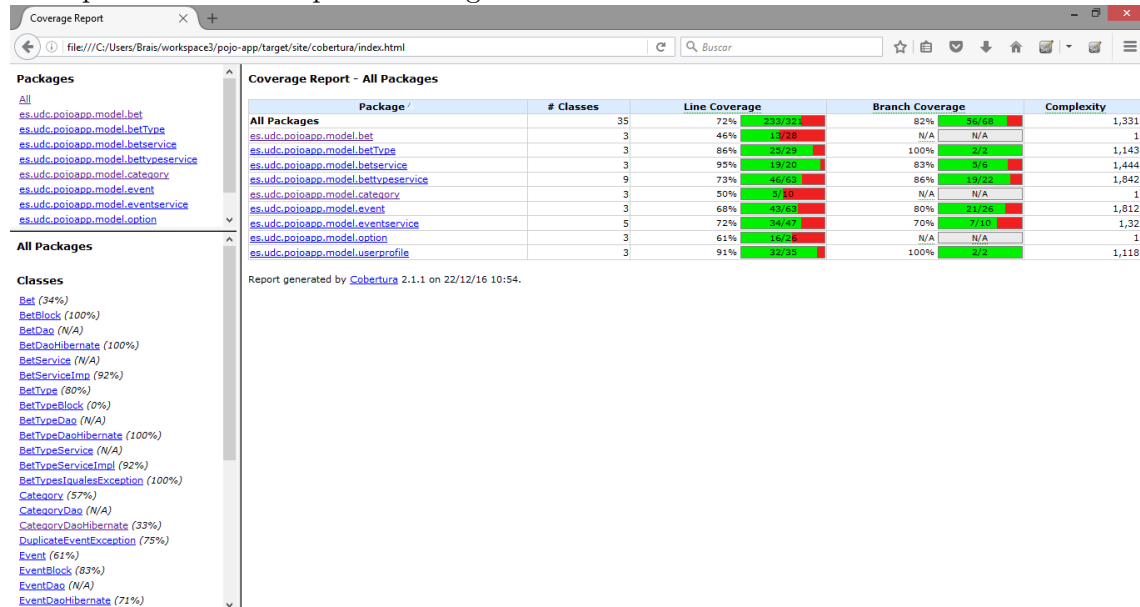
4.6. GraphWalker

GraphWalker realiza unha interfaz autoxenerada a partir de un grafo, *mvn graphwalker:generate-sources*, ao implementar esta interfaz xenerase un test para GraphWalker que pasa por todos os nodos do grafo. A execución do test realizase mediante o comando *mvn graphwalker:test*.

Abandonamos esta ferramenta debido a un error interno do propio GraphWalker.

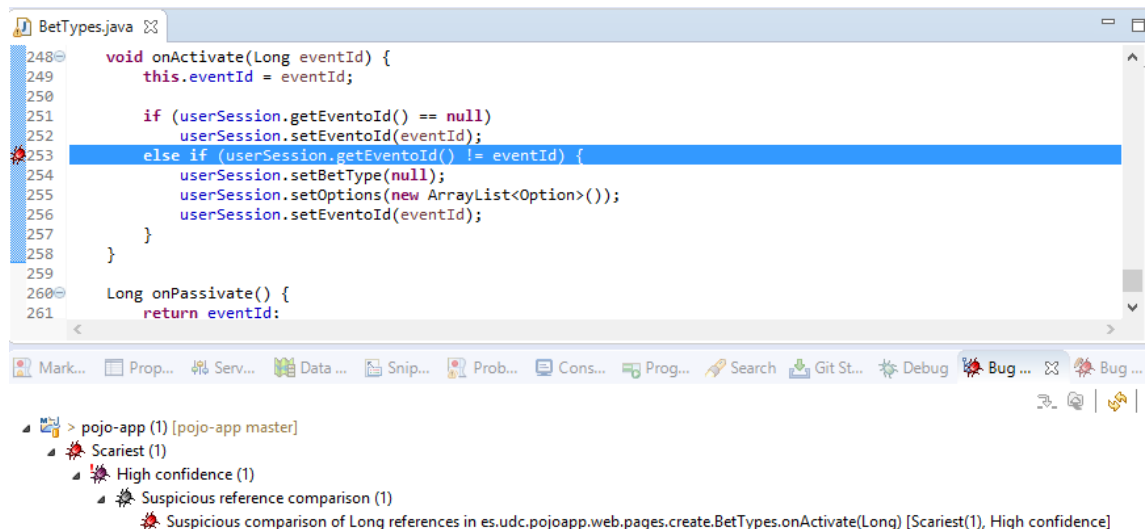
4.7. Cobertura

Utilizando `mvn cobertura:cobertura` no directorio raíz do proxecto, crease un reporte html coa cobertura total das clases e individualmente a de cada unha. Os datos deberían estar por encima do 65 por cento segundo os estándares software.



5. Rexistro de erros

5.1. FindBugs



6. Estatísticas

- *Número de erros encontrados diariamente e semanalmente: so encontramos un erro durante a realización das probas.*
- *Nivel de progreso na execución das probas: todas as probas de resultado positivo, exceptuando GraphWalker que rexeitamos debido a un erro da ferramenta.*
- *Análise do perfil de detección de erros: a maioría das probas centrase na parte do modelo da aplicación, mentras que as probas xeradas por JMeter e GraphWalker centranse na parte web.*
- *Informe de erros abertos e pechados por nivel de criticidade: Issue7: O erro foi atopado pola ferramenta FindBugs (foto a continuación), foi un erro de criticidade baixa , na categoría de boas practicas.*
- *Avaliación global do estado de calidade e estabilidade actuais: A calidade da aplicación permaneceu igual, xa que as probas nun detectaron ningun erro. Respecto ao estado inicial do proxecto aumentamos a cobertura de lineas probadas e aumentamos o número de probas sobre a parte web do proxecto.*
A única ferramenta que quedou sen probar foi GraphWalker debida a un erro da ferramenta.
En xeral as ferramentas deron un resultado positivo dando un proxecto de calidade.

- Estadísticas PITest:

Pit Test Coverage Report

Project Summary

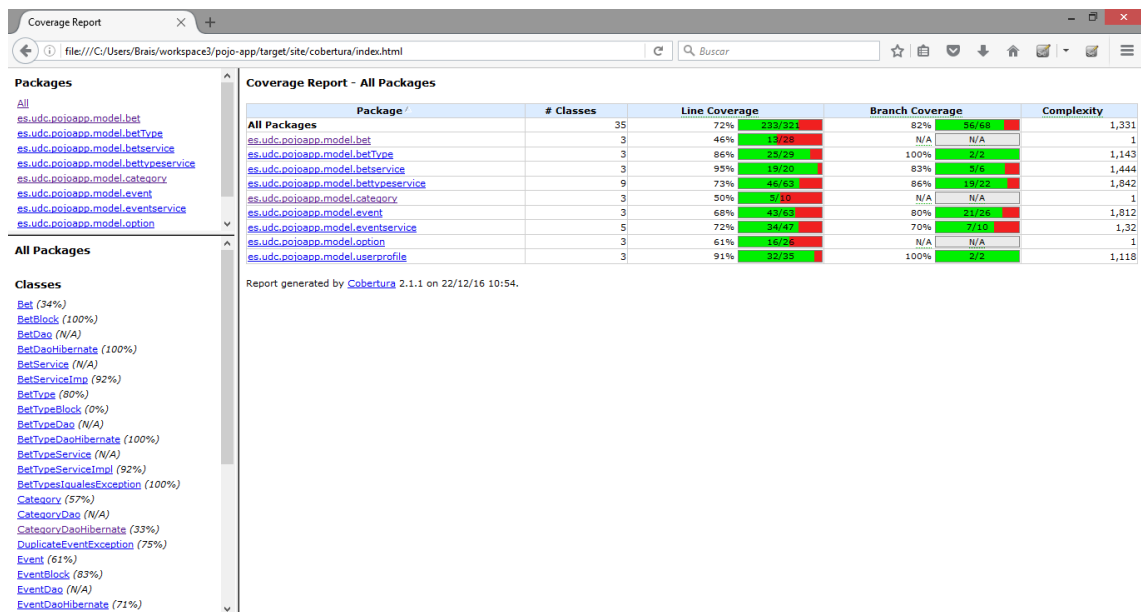
Number of Classes	Line Coverage	Mutation Coverage
20	72% <div><div></div></div> 224/310	69% <div><div></div></div> 86/124

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage
es.udc.pojoapp.model.bet	2	46% <div><div></div></div> 13/28	25% <div><div></div></div> 2/8
es.udc.pojoapp.model.betType	2	86% <div><div></div></div> 25/29	100% <div><div></div></div> 8/8
es.udc.pojoapp.model.betservice	2	95% <div><div></div></div> 19/20	100% <div><div></div></div> 12/12
es.udc.pojoapp.model.bettypeservice	3	72% <div><div></div></div> 38/53	76% <div><div></div></div> 19/25
es.udc.pojoapp.model.category	2	50% <div><div></div></div> 5/10	33% <div><div></div></div> 1/3
es.udc.pojoapp.model.event	2	68% <div><div></div></div> 43/63	68% <div><div></div></div> 19/28
es.udc.pojoapp.model.eventservice	4	72% <div><div></div></div> 34/47	56% <div><div></div></div> 14/25
es.udc.pojoapp.model.option	1	60% <div><div></div></div> 15/25	50% <div><div></div></div> 3/6
es.udc.pojoapp.model.userprofile	2	91% <div><div></div></div> 32/35	89% <div><div></div></div> 8/9

Report generated by [PIT](#) 1.0.0

- Estadísticas Cobertura :



■ Estadísticas JMeter :

